1功能、设计

2018年10月31日　　13:36

-
- 操作系统的主要功能

一. 处理机管理功能

1) 创建和撤消进程(线程)，对诸进程(线程)的运行进行协调，实现进程(线程) 之间的信息交换，以及按照一定的算法把处理机分配给进程(线程)

1. 进程控制：主要功能是为作业创建进程，撤消已结束的进程，以及控制进程在运行过程中的状态转换。在现代 OS 中，进程控制还应具有为一个进程创建若干个线程的功能和撤消(终止)已完成任务的线程的功能
2. 进程同步：为多个异步进程(含线程)的运行进行协调。有两种方式：
   1) 进程互斥方式：指访问临界资源时互斥
   2) 进程同步方式：指在相互合作去完成共同任务的诸进程间，由同步机构对它们的执行程序加以协调
      (1) 最简单的实现机制是为每一个临界资源配置一把锁 W，当锁打开时，进程(线程)可以对该临界资源进行访问；而当锁关上时，则禁止进程(线程)访问该临界资源
      (2) 最常用机制是信号量机制
3. 进程通信：相互合作的进程之间的信息交换
   1) 同一计算机系统内，通常是采用直接通信方式，即由源进程利用发送命令直接将消息(Message)挂到目标进程的消息队列上，以后由目标进程利用接收命令从其消息队列中取出消息
4. 调度
   1) 作业调度
      (1) 从后备队列中按照一定的算法，选择出若干个作业，为它们分配运行所需的资源(首先是分配内存)。在后备队列上等待的每个作业都需经过调度才能执行
      (2) 调入内存后，便分别为它们建立进程，使它们都成为可能获得处理机的就绪进程，并按一定的算法将它们插入就绪队列
   2) 进程调度
      (1) 从进程的就绪队列中，按照一定的算法选出一个进程，把处理机分配给它，并为它设置运行现场，使进程投入执行
      (2) 多线程 OS 中，通常把线程作为独立运行和分配处理机的基本单位。把就绪线程排成一个队列，每次调度时，从队列中选出一线程，分配处理机给它

二. 存储器管理功能

1) 提高存储器的利用率以及从逻辑上扩充内存

1. 内存分配：为每道程序分配内存空间，使它们“各得其所”；提高存储器利用率，以减少不可用的内存空间；允许正在运行的程序申请附加的内存空间，以适应程序和数据动态增长的需要
   1) 静态分配方式：每个作业的内存空间是在作业装入时确定的；在作业装入后的整个运行期间，不允许该作业再申请新的内存空间，也不允许作业在内存中“移动”
   2) 动态分配方式：每个作业所要求的基本内存空间也是在装入时确定的，但允许作业在运

行过程中继续申请新的附加内存空间，以适应程序和数据的动态增长，也允许作业在内存中“移动”

3) 内存分配机制里需要的结构和功能
   (1) 内存分配数据结构：记录内存空间的使用情况，作为分配依据
   (2) 内存分配功能：按一定的内存分配算法为用户程序分配内存空间
   (3) 内存回收功能：对于用户不再需要的内存，通过用户的释放请求去完成系统的回收功能

2. 内存保护：确保每道用户程序都只在自己的内存空间内运行，彼此互不干扰；绝不允许用户程序访问操作系统的程序和数据；也不允许用户程序转移到非共享的其它用户程序中去执行
   1) 一种比较简单的内存保护机制是设置两个界限寄存器，分别用于存放正在执行程序的上界和下界，系统须对每条指令所要访问的地址进行检查，如果发生越界，便发出越界中断请求，以停止该程序的执行。若用软件实现，则每执行一条指令，须增加若干条指令去进行越界检查，这将显著降低程序的运行速度。因此，越界检查都由硬件实现，对发生越界后的处理，还须与软件配合来完成

3. 地址映射
   1) 地址空间：地址所形成的地址范围，其中的地址称为“逻辑地址”或“相对地址”。程序的逻辑地址都从“0”开始的，程序中的其它地址都是相对于起始地址计算的
   2) 内存空间：内存中的一系列单元所限定的地址范围，其中的地址称为“物理地址”
   3) 为使程序能正确运行，存储器管理必须提供地址映射功能，以将地址空间中的逻辑地址转换为内存空间中与之对应的物理地址。 该功能应在硬件的支持下完成

4. 内存扩充：借助于虚拟存储技术，从逻辑上去扩充内存容量
   1) 请求调入功能：允许在装入一部分用户程序和数据的情况下，便能启动该程序运行。 在程序运行过程中，若发现要继续运行时所需的程序和数据尚未装入内存，可向 OS 发出请求，由 OS 从磁盘中将所需部分调入内存，以便继续运行
   2) 置换功能：若发现在内存中已无足够的空间来装入需要调入的程序和数据时，系统应能将内存中的一部分暂时不用的程序和数据调至盘上，以腾出内存空间，然后再将所需调入的部分装入内存

三. 设备管理功能

1) 完成用户进程提出的 I/O 请求；为用户进程分配其所需的 I/O 设备；提高 CPU 和 I/O 设备的利用率；提高 I/O 速度；方便用户使用 I/O 设备

1. 缓冲管理
   1) 在 I/O 设备和 CPU 之间引入缓冲，可有效地缓和 CPU 与 I/O 设备速度不匹配的矛盾，提高 CPU 的利用率， 进而提高系统吞吐量
   2) 最常见的缓冲区机制有单缓冲、双向同时传送的双缓冲，供多设备同时使用的公用缓冲池

2. 设备分配
   1) 根据用户进程的 I/O 请求、系统现有资源情况及某种设备的分配策略，为之分配其所需的设备
   2) 如果在 I/O 设备和 CPU 之间还存在着设备控制器和 I/O 通道，还须为分配出去的设备分配相应的控制器和通道
   3) 为了实现设备分配，系统中应设置设备控制表、控制器控制表等数据结构，用于记录设备及控制器的标识符和状态
   4) 在进行设备分配时，应针对不同的设备类型而采用不同的设备分配方式。对于独占设备(临界资源)，还应考虑到该设备被分配出去后系统是否安全。 在设备使用完后，应立即由系统回收

3. 设备处理
   1) 设备处理程序又称为设备驱动程序。其基本任务是用于实现 CPU 和设备控制器之间的通信，即由 CPU 向设备控制器发出 I/O 命令，要求它完成指定的 I/O 操作；反之，由 CPU 接收从控制器发来的中断请求，并给予迅速的响应和相应的处理

2) 处理过程是：设备处理程序首先检查 I/O 请求的合法性，了解设备状态是否空闲，了解有关的传递参数及设置设备的工作方式。然后向设备控制器发出 I/O 命令，启动 I/O 设备完成指定的 I/O 操作。设备驱动程序还应能及时响应由控制器发来的中断请求，并根据该中断请求的类型，调用相应的中断处理程序进行处理。对于设置了通道的计算机系统，设备处理程序还应能根据用户的 I/O 请求，自动地构成通道程序

四. 文件管理功能

1) 对用户文件和系统文件进行管理，方便用户使用，保证文件的安全性

1. 文件存储空间的管理：由文件系统对诸多文件及存储空间实施统一的管理。其主要任务是为每个文件分配必要的外存空间，提高外存的利用率，并能有助于提高文件系统的存、取速度

1) 系统应设置相应的数据结构，记录文件存储空间的使用情况，以供分配存储空间时参考；还应具有对存储空间进行分配和回收的功能。为了提高存储空间的利用率，对存储空间的分配，通常是采用离散分配方式，以减少外存零头，并以盘块为基本分配单位。盘块的大小通常为 1～8 KB。

2. 目录管理

1) 由系统为每个文件建立一个目录项，包括文件名、文件属性、文件在磁盘上的物理位置等
2) 由若干个目录项又可构成一个目录文件
3) 主要任务是为每个文件建立其目录项，并对众多的目录项加以有效的组织，以实现方便的按名存取，即用户只须提供文件名便可对该文件进行存取
4) 还应能实现文件共享，这样，只须在外存上保留一份该共享文件的副本
5) 还应能提供快速的目录查询手段，以提高对文件的检索速度

3. 文件的读/写管理和保护

1) 文件的读/写管理：该功能是根据用户的请求，从外存中读取数据，或将数据写入外存。在进行文件读(写)时，系统先根据用户给出的文件名去检索文件目录，从中获得文件在外存中的位置。然后，利用文件读(写)指针，对文件进行读(写)。一旦读(写)完成，便修改读(写)指针，为下一次读(写)做好准备。由于读和写操作不会同时进行，故可合用一个指针

2) 文件保护：为防止文件被非法窃取和破坏，必须提供有效的存取控制功能，以实现下述目标：

(1) 防止未经核准的用户存取文件
(2) 防止冒名顶替存取文件
(3) 防止以不正确的方式使用文件

五. 操作系统与用户之间的接口

1. 用户接口：提供给用户使用的接口，用户可通过该接口取得操作系统的服务

1) 联机用户接口：这是为联机用户提供的，它由一组键盘操作命令及命令解释程序所组成。当用户在终端或控制台上每键入一条命令后，系统便立即转入命令解释程序，对该命令加以解释并执行该命令。在完成指定功能后，控制又返回到终端或控制台上，等待用户键入下一条命令。这样，用户可通过先后键入不同命令的方式，来实现对作业的控制，直至作业完成

2) 脱机用户接口/批处理用户接口：该接口是为批处理作业的用户提供的，由一组作业控制语言(JCL)组成。批处理作业的用户不能直接与自己的作业交互作用， 只能委托系统代替用户对作业进行控制和干预。JCL便是提供给批处理作业用户的、为实现所需功能而委托系统代为控制的一种语言。用户用 JCL 把需要对作业进行的控制和干预事先写在作业说明书上，然后将作业连同作业说明书一起提供给系统。 当系统调度到该作业运行时，又调用命令解释程序，对作业说明书上的命令逐条地解释执行，直至遇到作业结束语句时，系统才停止该作业的运行

3) 图形用户接口：用户可用鼠标或通过菜单和对话框来完成对应用程序和文件的操作。用户从繁琐且单调的操作中解脱了出来。 图形用户接口方便地将文字、图形和图像集成在一个文件中。可以在文字型文件中加入一幅或多幅彩色图画，也可以在图画中写入必要的文字，而且还可进一步将图画、文字和声音集成在一起。20 世纪 90 年代以后推出的主流 OS 都提供了图形用户接口

2. 程序接口：提供给程序员在编程时使用的接口，是用户程序取得操作系统服务的唯一途径

1) 由一组系统调用组成，每个系统调用都是一个能完成特定功能的子程序，每当应用程序要求 OS 提供某种服务(功能)时，便调用具有相应功能的系统调用。早期的系统调用都是汇编语言提供的，只有在用汇编语言书写的程序中才能直接使用系统调用；但在高级语言以及 C 语言中，往往提供了与各系统调用一一对应的库函数，这样，应用程序便可通过调用对应的库函数来使用系统调用。但在近几年所推出的操作系统中，如 UNIX、OS/2 版本中，其系统调用本身已经采用 C 语言编写，并以函数形式提供，故在用 C 语言编制的程序中，可直接使用系统调用

六. 现代操作系统的新功能

1. 系统安全

1) 认证技术：通过一些参数的真实性来确认对象是否名副其实
2) 密码技术：为存储和传输的数据加密
3) 访问控制技术：设置用户存取权限、设置文件属性
4) 反病毒技术：预防、及时扫描可执行文件，清除病毒

2. 网络功能和服务

1) 网络通信：建立和拆除通信链路、传输控制、差错控制、流量控制等
2) 资源管理：协调硬、软件共享资源
3) 应用互操作：互联网络中实现信息的互通性和消息的互用性

3. 支持多媒体

1) 接纳控制功能
   (1) 限制软实时任务数、驻留在内存中的任务数
   (2) 媒体服务器、存储器、进程都设置了相应的接纳控制功能
2) 实时调度
   (1) 考虑进程调度策略的同时还要考虑进程调度的接纳度
   (2) 如图像更新周期应在40ms内
3) 多媒体文件的存储
   (1) 尽量把硬盘上的数据快速转到输出设备上
   (2) 因而需要改进数据离散存放方式和磁盘寻道方式

-
- OS结构设计

一. 传统的操作系统结构

1. 无结构操作系统/整体系统结构

1) 早期设计者只把注意力放在功能的实现和获得高的效率上，此时的 OS 是为数众多的一组过程的集合，每个过程可以任意地相互调用其它过程，致使操作系统内部既复杂又混乱，调试工作带来很多困难；另一方面也使程序难以阅读和理解，增加了维护人员的负担

2. 模块化结构OS

1) 模块化程序设计技术是 20 世纪 60 年代出现的一种结构化程序设计技术。该技术是基于“分解”和“模块化”原则来控制大型软件的复杂度。使各模块之间能通过该接口实现交互。然后，再进一步将各模块细分为若干个具有一定功能的子模块，这种设计方法称为模块—接口法

2)

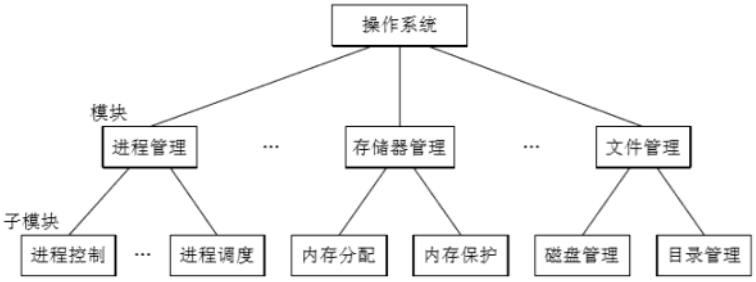


图 1-6 模块化结构的操作系统

3) 模块的独立性
   (1) 内聚性：指模块内部各部分间联系的紧密程度。
      i. 内聚性越高，模块的独立性越强

(2) 耦合度：指模块间相互联系和相互影响的程度。

i. 耦合度越低，模块的独立性越好

4) 模块接口法/无序模块法的优缺点

(1) 提高 OS 设计的正确性、可理解性和可维护性

(2) 增强 OS 的适应性

(3) 加速 OS 的开发过程

(1) 在 OS 设计时，对各模块间的接口规定很难满足在模块完成后对接口的实际需求

(2) 在 OS 设计阶段，设计者必须做出一系列的决定(决策)，每一个决定必须建立在上一个决定的基础上。但在模块化结构设计中，各模块的设计齐头并进，造成各种决定的"无序性"

3. 分层式结构OS

1) 自底向上的分层设计的基本原则是：每一步设计都是建立在可靠的基础上， 每一层仅能使用其底层所提供的功能和服务，每层又由若干个模块组成， 各层之间只存在着单向的依赖关系，即高层仅依赖于紧邻它的低层

2) 分层结构的优缺点

(1) 易保证系统的正确性

(2) 易扩充和易维护性，不修改接口就不影响其他层

(1) 系统效率低，增加了系统的通信开销

二. 客户/服务器模式(Client/Server)

1. 客户/服务器模式模式的组成

1) 客户机：可发送一个消息给服务器，以请求某项服务，平时它处理一些本地业务

2) 服务器：驻留有网络文件系统或数据库系统等， 它应能为网上所有的用户提供一种或多种服务。平时它一直处于工作状态，被动地等待来自客户机的请求，并将结果送回客户

3) 网络系统：用于连接所有客户机和服务器，实现它们之间通信和网络资源共享的系统

2. 客户/服务器之间的交互

1) 客户发送请求消息：发送进程先把命令和有关参数装配成请求消息，然后把它发往服务器；接收进程则等待接收从服务器发回来的响应消息

2) 服务器接收消息：接收进程平时处于等待状态，根据请求信息的内容，将之提供给服务器上的相应软件进行处理

3) 服务器回送消息：软件根据请求进行处理，在完成指定的处理后，把处理结果装配成一个响应消息，由发送进程发往客户机

4) 客户机接收消息：接收进程把收到的响应消息转交给软件，再由后者做出适当处理后提交给发送该请求的客户

3. 客户/服务器模式的优缺点

1) 数据的分布处理和存储

2) 便于集中管理

3) 灵活性和可扩充性

4) 易于改编应用软件

1) 存在着不可靠性和瓶颈问题。一旦服务器故障，将导致整个网络瘫痪。当服务器在重负荷下工作时，会因忙不过来而显著地延长对用户请求的响应时间。如果在网络中配置多个服务器，并采取相应的安全措施，则可加以改善

三. 面向对象的程序设计

1. "重用"提高产品质量和生产率； 具有更好的易修改性和易扩展性；更易于保证系统的"正确性"和"可靠性"

四. 微内核OS结构

1. 微内核操作系统的基本概念

(1) 为了提高操作系统的正确性、灵活性、易维护性和可扩充性：

1) 足够小的内核：

(1) 实现与硬件紧密相关的处理

(2) 实现一些较基本的功能

(3) 负责客户和服务器之间的通信

2) 基于客户/服务器模式

(1) 最基本的部分放入内核中，而把绝大部分功能都放在微内核外面的一组服务器(进程)中实现

(2)

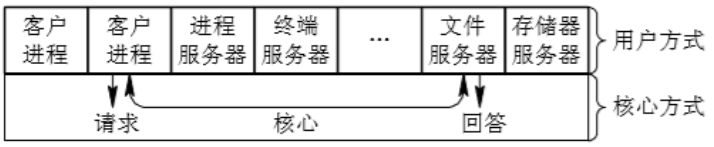


图 1-10 在单机环境下的客户/服务器模式

3) 应用“机制与策略分离”原理

1) 机制，是指实现某一功能的具体执行机构
2) 策略，是在机制的基础上，借助于某些参数和算法来实现该功能的优化
3) 通常，机制处于系统的基层（微内核），而策略则处于高层

4) 采用面向对象技术

(1) “抽象”和“隐蔽”原则控制系统的复杂性，“对象”、“封装”和“继承”等概念来确保操作系统的“正确性”、“可靠性”、“易修改性”、“易扩展性”等

2. 微内核的基本功能

1) 进程(线程)管理：在进程管理中设置一个或多个进程(线程)优先级队列； 能将指定优先级进程(线程)从所在队列中取出，并将其投入执行。由于这一部分属于调度功能的机制部分，应将它放入微内核中。应如何确定每类用户(进程)的优先级，以及应如何修改它们的优先级等，都属于策略问题，可将它们放入微内核外的进程(线程)管理服务器中。

由于进程(线程)之间的通信功能是微内核 OS 最基本的功能，被频繁使用，因此几乎所有的微内核 OS 都是将进程(线程)之间的通信功能放入微内核中。此外,还将进程的切换、线程的调度，以及多处理机之间的同步等功能也放入微内核中

2) 低级存储器管理：如用于实现将用户空间的逻辑地址变换为内存空间的物理地址的页表机制和地址变换机制，这部分依赖于机器，因此放入微内核。而实现虚拟存储器管理的策略，则包含应采取何种页面置换算法，采用何种内存分配与回收策略等，应放在微内核外的存储器管理服务器中

3) 中断和陷入处理：主要功能是捕获所发生的中断和陷入事件，并进行相应的前期处理。如进行中断现场保护，识别中断和陷入的类型，然后将有关事件的信息转换成消息后，把它发送给相关的服务器。由服务器根据中断或陷入的类型，调用相应的处理程序来进行后期处理。 在微内核 OS 中是将进程管理、存储器管理以及 I/O 管理这些功能一分为二，属于机制的很小一部分放入微内核中

3. 微内核操作系统的优点

1) 提高了系统的可扩展性（由于微内核 OS 许多功能是由相对独立的服务器软件实现的）

2) 增强了系统的可靠性

(1) 微内核是出于精心设计和严格测试的容易保证其正确性
(2) 提供了规范而精简的应用程序接口(API)，为微内核外的高质量程序代码创造了条件
(3) 所有服务器都是运行在用户态，服务器与服务器之间采用的是消息传递通信机制，服务器出现错误时，不会影响内核，也不会影响其它服务器

3) 可移植性（所有与特定 CPU 和 I/O 设备硬件有关的代码，均放在内核和内核下面的硬件隐藏层中， 而操作系统其它绝大部分均与硬件平台无关）

4) 提供了对分布式系统的支持

(1) 客户和服务器之间以及服务器和服务器之间的通信，是采用消息传递通信机制进行的，致使微内核 OS 能很好地支持分布式系统和网络系统
(2) 只要在分布式系统中赋予所有进程和服务器惟一的标识符，在微内核中再配置一张系统映射表 (进程和服务器的标识符与它们所驻留的机器之间的对应表)，在进行客户与服务器通信时，只需在所发送的消息中标上发送进程和接收进程的标识符，微内核便可发送消息

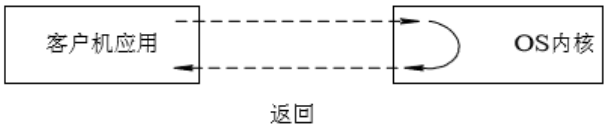
5) 融入了面向对象技术

(1) “封装”，“继承”，“类”和“多态”，以及对象间采用消息传递机制等，都有利于提高系统“正确性”、“可靠性”、“易修改性”、“易扩展性”等，还能显著减少开发系统所付出的开销

4. 微内核操作系统存在的问题

1) 运行效率有所降低：在完成一次客户对 OS 提出的服务请求时，需要利用消息实现多次交互

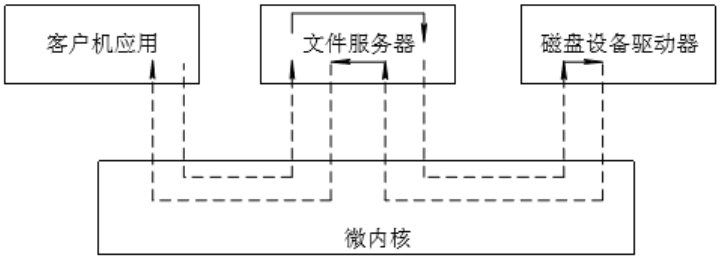
和进行用户/内核模式及上下文的多次切换。然而，在早期的 OS 中，用户进程在请求取得 OS 服务时，一般只需进行两次上下文的切换。如图 1-11 中所示，其中的文件服务器还需要磁盘服务器的帮助，这时就需要进行八次上下文的切换



返回

(a) 在整体式内核文件操作中的上下文切换

2)



(b) 在微内核中等价操作的上下文切换

3) 为改善运行效率，可以重新把一些常用的基本功能由服务器移入微内核中。但这又会使微内核的容量明显地增大，在小型接口定义和适应性方面的优点也有所下降，也提高了微内核的设计代价

i.
ii.
iii.
iv.
v.
vi. --------我是底线----------